DATE: 4/02/97

CONTROLLING OFFICE FOR THIS DOCUMENT IS:
Department of the Army
Office of the Director of Information Systems
for Command, Control, Communications and
Computers (ODISC4)
The Pentagon
Washington, DC 20310-0600

POC: ODISC4

DISTRIBUTION STATEMENT A: Public release

June 1996

Defusing the Millennium Time Bomb

Scale and scope of Defense Department systems create management and coordination challenges.

By Andrew C. Braunberg

A worldwide computer time bomb is ticking toward a cataclysmic explosion. This detonation could cause total collapse in international business, financial and defense markets unless partially identified hardware and software corrections can be implemented on an unprecedented scale.

Without attention, almost all of the world's microprocessor-dependent systems progressively will become unreliable between now and the year 2000. The global market for correcting the problem could total $200 billion to $400 billion by the year 2000, according to a widely touted estimate.

An independent MITRE Corporation assessment of military vulnerability to the software problem describes it as real and potentially catastrophic. Military logistics systems, in the process of creating five-year stockpiles, already are encountering the problem, as are financial and insurance institutions. Critical systems in the U.S. Defense Department could face substantial failure if the department does not aggressively address what is known as the year 2000 problem.

The problem arises from the once-common practice of representing years by only their last two digits, for example, 1996 as 96 and 2000 as 00. While the year 2000 obviously occurs after the year 1996, two-digit representations change this relationship. This is because computers use mathematical comparisons to determine time sequence. A computer employing two-digit year representations would determine incorrectly that the year 00 occurs before the year 99 when a simple greater-than comparison is done. In other words, because 99 is greater than 00, the year 99, or 1999, is calculated to have come after the year 00, or 2000. A host of mistakes can result from two-digit year representations.

The MITRE report estimates a cost of between $1 and $8 per line for the more than 1 billion lines of code in various Defense Department systems. The department also may have to initiate a microprocessor production line to manufacture replacement chips for firmware-dependent weapons systems that no longer perform date calculations properly.

The problem mimics a virus. The flaw will not stop a machine from processing but could induce incorrect calculations. While technical solutions are simple and well understood, the Defense Department's task is compounded by the complexity of its many systems, the use of obscure and unpopular programming languages and uncertainty in the total amount of military software.

Tool vendors are addressing the problem and creating programs to automate or semi-automate the correction. Each tool is written for a specific programming language. Approximately 80 percent of the worldwide market is for COBOL applications at insurance companies and banks, according to Tom Backman, director, software engineering center, MITRE Corporation. He adds that 85 percent to 95 percent of the tool vendors currently are dealing with software for large main frames.

Insurance and financial institutions need to address the problem sooner than most other organizations because of their long-term horizons. For example, insurance companies calculate actuaries of more than 100 years, and financial firms commonly make 30-year loans. Defense-related occurrences have been limited.

Tool vendors are following the early market in insurance and finance. This is concentrating the tool base in COBOL--and some personal computer--applications.

Many Defense Department systems are written in COBOL, but most military command and control systems and sensor systems are not, Backman observes. More likely, these programs either were written in Ada, C, Assembler or used firmware chips.

The Defense Department conceivably might have to start up a chip production line because microprocessors in missiles no longer perform date calculations properly, according to Robert Molter, computer scientist, information technology directorate, command, control, communications and intelligence, Department of Defense. He emphasizes that, while the military has not found such faulty chips, it is checking all systems.

Molter, who is also chairman of the Defense Department's year 2000 working group and a member of the interagency year 2000 committee, states that the most difficult part of the problem is solving the interconnections between systems. Tremendous ramifications exist concerning the corruption of data bases, he contends, adding that the department needs to work on strategies to isolate or eliminate problem areas. Safety concerns also exist. Molter notes that weapons systems and systems that affect safety are receiving priority treatment.

There is not a wealth of tools to address the year 2000 problem for most software languages. Molter allows that languages specifically are a problem for the department. Even COBOL code is problematic because the department has a lot of old COBOL, which does not support year 2000 tools. The department will have to upgrade to newer versions of COBOL, according to Molter. He adds that Ada is one language that is not a big concern. If programmers use the date typing within Ada correctly, the change of century will not pose a problem. The code still has to be checked, though, Molter admits.

The lack of available tools to automate the process could increase personnel requirements. Making the actual correction is not the only, or the most significant, cost related to the problem, Backman laments. Testing systems and ensuring interoperability are costly and difficult tasks. Backman suggests that the major concerns are in management and coordination, not in technology.

For example, updating the code in a radar system may require changing only several lines of code. Testing the radar, however, could include several aircraft performing flybys. Not all systems need to be changed, but they all need to be analyzed, Backman states. Cost savings are possible if corrections concur with general system maintenance. For example, programmers could take advantage of scheduled testing and maintenance exercises to make required year 2000 changes. Unmaintained systems, he notes, will require much higher costs because of a lack of scheduled testing.

The optimal solution depends on user needs and interoperability requirements. For example, three interconnected users all could change their computer codes to four-digit compatibility--the most comprehensive and costly solution--or one of the three users could be designated as the date keeper. As such, that one user could make corrections and then feed the accurate data to the other two users.

No universal cost estimates exist for performing the software fix, Backman states. Estimators need to know where the system is in its normal maintenance cycle, how large it is and what correction should be made. Because all of these parameters affect costs and because this information is difficult to obtain, particularly the size of systems, hard cost numbers are immediately suspicious.

Given continuing budget cuts, the problem is especially untimely. The issue was unanticipated and remains unbudgeted. Some areas are scrambling to figure out how to address the situation, Molter admits. The department needs a data base of information with a baseline of systems, including all hardware and software technical environments. An inventory of all the software in the department has yet to be completed, Molter notes, estimating that the total will be in excess of 1 billion lines of code, all of which must be checked. This type of inspection will redirect people who would have been developing new systems and adding features, Molter notes. The department will have to divert personnel for the next three years to address the problem.

MITRE did a better job than other organizations on cost projections, according to Molter. The corporation predicts that changes to automated information systems could cost as little as $1 per line of code, while changes to command and control systems could cost as much as $8.52 per line of code. Although not absolute, Molter allows that these estimates are certainly "in the ballpark." Given the probable amount of code involved, $1 billion in total costs is a conservative estimate, he states.

Tools are available to semi-automate the process of software correction. But the military market has only a small subset of the tools available to the commercial market. Two of the more popular commercial tools are slicers and parsers.

A parser reads code, extracts variables and assignment statements from it and then converts the code into a form compilers can work with. A parser separates equal signs, multiplication and logic statements. This creates a code that is easier to examine, Backman states.

Slicing tools can scan code to find all occurrences of dates and all references to them. This allows programmers to localize where a problem may reside. A slicer can flag these

spots so that a programmer can do post-analysis on these areas. A human is required to determine, for any given application, what the least costly fix to the problem is. For example, a programmer should know the range of date data. A program for performing four-year forecasts might be able to build a sliding window to solve the problem. A sliding window could retain the two-digit year by creating a software fix to bypass the problem.

Before the code can be examined, it must be located. Backman relates that, during a six-week study of the problem, MITRE could not find an inventory of all military information systems. He adds that the team also was unsuccessful in determining the total number of lines of code contained in defense systems. Backman will not even hazard a guess. Finding documentation is an additional chore, especially for unmaintained code.

Tools are needed for locating all of the software code in the environment. Users then can search for error code. Programmers need to look at all the code that operates on dates. All if-then-else, comparison and assignment statements should be parsed out. The impact to each of these statements is dependent on the application.

Existing tools are efficient at finding possible problem locations. Humans still are needed to ascertain the impact of the problem for each application and to work with other code users to determine the best fix. Users also need to verify the integrity of data from commercial packages. Commercial software has embedded date code that could affect other code.

Not all date problems reside in software. Firmware chips also can contain date information. Firmware chips often are written in microcode, and programmers may no longer have the cross compiler that was used to put the system together.

A four-digit fix is preferred but not necessarily cost effective. Backman warns against mandating four-digit year compliance, noting that each community of interest will know the best local solution. He recommends decentralizing execution while demanding interoperability where needed. He adds that the Defense Department should centralize information dissemination and collection.

Most commercial industries are deferring corrections until 1998, Backman allows. He adds that, because the Defense Department is so large, has so many systems and interacts with so many contractors, it cannot wait until 1998. This time constraint has both benefits and costs, Backman notes. The earlier the problem is addressed, the less expensive and more effective the solution. But piggybacking the correction into general maintenance may require a large time window depending on the maintenance cycle. Updated commercial software can be replaced. But, to take advantage of software changes, such as in operating systems, vendors may require users to update hardware also. The military has a history of slightly out-of-date infrastructure hardware. Often, hardware can be at most two or three years old to run new operating systems. Military hardware that is 10 years old is not uncommon, Backman reports. Users need to know what operating system a computer is running and what software release can fix the problem.

The year 2000 problem creates security concerns also, according to Backman. Many system administrators turn system passwords on and off. For example, access might be denied to certain users on weekends. By inverting time comparisons, the problem could deny access to legitimate users while allowing access to hackers. This administrative example highlights a type of problem that could be very serious, Backman states.

Additional information is available on the World Wide Web at Navy http://www.nismc.navy.mil; Air Force http://infosphere.safb.af.mil (A project home page exists for the year 2000 problem at this site.); Armyhttp://iiacsun1.army.mil; and DISA http://www.disa.mil/line/jieojex.html.

Back to June 1996 's Home Page

Back to *SIGNAL*'s Home Page